International Journal of Communication Networks and Information Security
2022, 14(03),
ISSN: 2073-607X,2076-0930
https://https://ijcnis.org/

Research Article

Machine Learning for Anomaly Detection in the Financial Close: A Journal Entry Risk-Scoring Framework for SAP S/4HANA

Rahul Rao Juvvadi

CPA, CGMA
rahulraoj99@gmail.com

Abstract

Manual journal entries (JEs) posted during the financial close have long been treated as a residual source of fraud and error, and neither rule-based controls nor statistical sampling deals with them well. This paper sets out a machine learning (ML) risk-scoring framework for SAP S/4HANA. It reads journal data straight from the Universal Journal table (ACDOCA), builds features around four risk constructs — anomalous posting behavior, unusual general-ledger (GL) account combinations, reversal activity, and round-amount transactions — and returns a risk score for each entry to the close-review workflow. Unsupervised anomaly detection (an Isolation Forest and a shallow autoencoder) is paired with a supervised gradient-boosted classifier trained on entries that earlier audits had flagged, and the three are fused into a hybrid ensemble. On roughly 2.5 million entries from a single company code in a multinational manufacturer's SAP S/4HANA 2021 tenant, spanning twelve fiscal periods, the ensemble reached precision 0.94, recall 0.91, F1 0.92, ROC-AUC 0.97, and PR-AUC 0.89 on the held-out test set. We report precision@k alongside the false-positive rate (FPR) because, at the operating point quoted above, the raw count of false positives outruns any realistic review capacity at this prevalence. The framework is meant to support, not replace, the auditor's required journal entry procedures under PCAOB AS 2401 [1], and to enable continuous controls monitoring (CCM) in the spirit of the COSO 2013 framework [26]. We also work through SHAP explainability one model family at a time, the label bias inherited from prior audit findings, and the IT general control (ITGC) questions that SOX §404 raises about the scoring model itself.

Keywords *— SAP S/4HANA; ACDOCA; journal entry testing; anomaly detection; continuous auditing; SOX §404; SHAP.*

I. Introduction

Few processes in enterprise accounting carry as much risk as the financial close. The final adjusting and accrual entries are often made by hand, against a deadline, and they move through controls that were really designed for routine, high-volume postings. Manual journal entries (JEs) are a well-understood fraud vector, and external auditors have scrutinized them for decades. For issuers, the Public Company Accounting Oversight Board (PCAOB) requires auditors to examine journal entries and other adjustments for evidence of possible material

Copyright © 2024 by Author/s and Licensed by IJCNIS. This is an open access article distributed under the Creative Commons Attribution License which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

misstatement due to fraud (AS 2401, paragraphs .58–.62) [1]; auditors of non-issuers work under a parallel requirement in AICPA AU-C 240 [2]. Both management's assertion on the effectiveness of internal control over financial reporting (ICFR) under §404 of the Sarbanes-Oxley Act and the auditor's attestation on it under PCAOB AS 2201 ultimately rest on whether JE controls are well designed and actually operating [3].

The controls most shops lean on — segregation-of-duties enforcement in SAP Governance, Risk and Compliance (GRC) Process Control, monetary thresholds, weekend-posting flags, and manual sampling — throw off a high rate of false positives on routine entries yet still miss the quieter problems: collusion across user IDs, a posting time that depends on the document type, or reversals timed to manage interim balances. For precisely those subtler patterns, the fraud-detection literature has argued for years that statistical and machine-learning signals read off the transaction stream beat fixed-threshold rules [4]–[6]. Continuous auditing and continuous controls monitoring were proposed three decades ago [7] and have been built out a good deal since [8], [9]; even so, they remain hard to operate, because rule-based methods simply do not scale to the entry-by-entry review that vision assumes.

What we propose is a machine learning risk-scoring framework built on the SAP S/4HANA Universal Journal (ACDOCA). It turns the four risk-construct families into a per-entry feature vector, scores every entry through both unsupervised and supervised models, and blends those outputs in a hybrid ensemble. The score then flows back into the SAP close workflow, so reviewers can decide which entries to put forward for substantive testing first.

Our contributions are:

1. A feature design grounded in ACDOCA field semantics and the AS 2401 journal entry testing requirements.
2. A hybrid ensemble that addresses two well-known problems — sparse labels and concept drift in accounting populations — by combining unsupervised anomaly detection with supervised classification on prior audit findings.
3. An explainability layer using model-appropriate SHAP variants: TreeSHAP for the gradient-boosted classifier, KernelSHAP on reconstruction-error attributions for the autoencoder, and path-length attributions for the Isolation Forest.
4. A governance discussion that explicitly treats the scoring model itself as in scope for SOX §404 ITGC testing.

There is, of course, existing SAP-native tooling: SAP GRC Process Control, SAP Audit Management, the legacy SAP Audit Information System (AIS), and the standard ACDOCA-based SoD controls. These give you rule-based exception detection and audit-workflow management, but none of them learns from past audit outcomes or scores an individual entry by composite risk. Our framework sits next to them rather than over them — GRC enforces preventive segregation-of-duties, while the framework directs detective review effort to where it pays off.

The rest of the paper is organized as follows. Section II reviews prior work. Section III lays out the SAP environment, the feature design, and the data flow. Section IV turns to model development, evaluation, and SHAP interpretation. Section V deals with governance, ITGC, and continuous monitoring, and Section VI closes with limitations and future work.

II. Related Work

A. Journal Entry Fraud Detection and Risk Indicators

The four risk constructs we operationalize — posting behavior, unusual account combinations, reversal patterns, and round-amount transactions — appear consistently in the fraud-detection literature. Beneish's M-Score uses accruals and accrual-quality variables to detect earnings manipulation [4]; Dechow et al.'s F-Score extends this to a probability of misstatement using publicly available financial-statement inputs [5]. At the transaction level, Debreceny and Gray applied digital analysis, including Benford's Law on leading-digit distributions, to journal entry populations and identified deviations associated with manipulation [6]. Bertomeu et al. found that machine learning models incorporating accounting variables, governance, and audit signals outperform traditional discriminant scores for misstatement detection [10]. Guo, Yu, and Wilkin treat the JE population as a graph of account-account and account-user relationships, arguing that topology-aware features expose patterns that row-level controls miss [11]. The "Sherlock" framework of Bay et al. is an early but instructive example of ledger-wide irregularity flagging integrated with auditor workflow [12].

Ding et al. sharpened actuarial estimates against expert benchmarks for insurance loss reserves [13]. That result is a particular case of a more general pattern: once past outcomes have been labeled, supervised learning tends to match or beat expert judgment on the routine cases, which frees the expert to spend time on the genuinely unusual ones.

B. Anomaly Detection Methods Applied to Accounting and Log Data

Hemati, Schreyer, and Borth proposed continual learning for unsupervised anomaly detection in journal entry data, which speaks directly to the concept-drift problem that recurs from one reporting period to the next [14]. An earlier study by Schreyer et al. had already put deep autoencoder networks to work on large-scale accounting data, scoring entries by reconstruction error [15]; that work is part of why we keep an autoencoder branch. Oliveira and Sousa, meanwhile, ran k-means clustering for retail anomaly detection inside SAP HANA XS Advanced, a useful demonstration of doing the unsupervised work in-platform [16].

Look to adjacent fields and the same ideas keep working. Unsupervised and semi-supervised deep models have done well on event logs and timelines [17], [18]; AutoLog, a semi-supervised deep autoencoder for system logs, beat both Isolation Forest and one-class SVM baselines on industrial data [18]. LSTM sequence models have caught abnormal patterns in time-structured logs [19], and stacking an unsupervised detector beneath a supervised classifier has been shown to improve detection of both familiar and previously unseen attacks in intrusion detection [20], [21]. It is that stacking idea we borrow for the ensemble.

C. Explainability and Governance

When the stakes run this high, a controller or an external auditor has to be able to read what the model is saying. Antwarg et al. showed how to explain an autoencoder's anomaly scores with SHAP, attributing reconstruction error back to the input features [22]; the variants we use all rest on Lundberg and Lee's original SHAP work [23] (TreeSHAP for tree ensembles, with KernelSHAP and DeepSHAP as model-agnostic fallbacks). Farrugia et al. combined predictive scoring with local interpretation and drift management in a fraud-detection setting much like

ours [24]. And Balayn et al. survey bias and fairness in data-driven decision support [25] — which matters here, because audit-derived labels carry the auditors' own attention bias, a point we come back to in Section IV.

D. Continuous Auditing in ERP Environments

The conceptual groundwork was laid twice over: by Vasarhelyi and Halper in their original work on continuous online audit [7], and by Alles, Kogan, and Vasarhelyi in their analysis of the economics of continuous assurance [8]. Kuhn and Sutton later surveyed continuous auditing in ERP settings specifically and named the gaps this framework begins to close — scoring at the level of the individual entry, feedback that lives inside the system rather than in a parallel report, and a genuine tie-in to the close workflow [9].

III. Framework Design

A. SAP S/4HANA Financial Close Environment

The Universal Journal — table ACDOCA — arrived as a single line-item table covering general ledger, controlling, asset accounting, and material ledger postings at once. Every row already carries the full set of financial and management-accounting dimensions, so the reconciliations that used to run between separate tables simply disappear. That is what makes entry-level ML scoring practical here: the dimensions you need sit in one place, reachable through standard ABAP Core Data Services (CDS) views.

We target SAP S/4HANA 2021, and the design holds for any pre-2022 on-premise or private-cloud release. Routine postings — automated billing journals, say — already clear GRC Process Control and the standard SoD checks. The residual exposure sits with the manual entries: accruals, reclassifications, and top-side adjustments. Those are what this framework is built to watch.

B. Feature Engineering and ACDOCA Field Mapping

Feature engineering is what turns raw ACDOCA rows into something the model can actually score. Table I lists the features we used in the experiments, with the ACDOCA field references corrected. Two of those fields deserve a comment:

- **Document entry time vs. posting date.** ACDOCA-CPUTM is the *document entry time* (timestamp at which the document was created in the system), and ACDOCA-CPUDT is the *document entry date*. The *posting date* — the accounting date for which the entry takes effect — is ACDOCA-BUDAT. Risk signals derived from "after-hours posting" mix these two semantics; we therefore separate them into *entry timestamp* (BUDAT, CPUDT, and CPUTM combined) and *posting-period proximity to close* (POPER vs. BUDAT relative to period-end).
- **Reversal indicators.** ACDOCA-XREVERSAL flags the document as a *reversing* document; ACDOCA-XREVERSED flags an entry that *has been* reversed. Both carry signal, and we use both.

On top of those we bring in — and used in the experiments — the document type (BLART) and the document status (BSTAT). BLART separates manual-entry document types (SA, the G/L

account document, for instance) from automated ones (RV for billing), which makes it a strong risk covariate. BSTAT tells parked from posted from reversed. We also tried the reference procedure / object type (AWTYP); it did not help in our setting, so we dropped it, though we note it here for anyone looking to replicate the study.

Table I. Journal Entry Features Used in the ML Framework

Feature	Data Type	Risk Construct	ACDOCA Field(s)
Entry timestamp	Datetime	Posting behavior	BUDAT + CPUDT + CPUTM
User ID	Categorical	Posting behavior	USNAM
Document type	Categorical	Posting behavior	BLART
GL account	Categorical	Account combination	RACCT
GL-account co-occurrence	Categorical (engineered)	Account combination	derived from RACCT pairs within BELNR
Amount in local currency	Numeric	Round-amount / size	HSL
Round-amount flag	Binary (engineered)	Round-amount	derived from HSL
Reversing-doc flag	Binary	Reversal pattern	XREVERSAL
Reversed-doc flag	Binary	Reversal pattern	XREVERSED
Posting-period proximity to close	Numeric (engineered)	Posting behavior	POPER + BUDAT

The six constructs of the original design now appear as ten engineered features in Table I, still grouped under the four headings used throughout the paper: posting behavior, unusual account combinations, reversal patterns, and round-amount transactions.

C. Data Flow Architecture

Extraction runs off ABAP CDS views with analytical annotations over ACDOCA, surfaced to SAP Data Intelligence through the SAP CDS Reader operator. SAP Data Intelligence then orchestrates the export into Azure Data Lake Storage Gen2 as Parquet, partitioned by company code and fiscal year/period. Training happens in Azure Machine Learning (v1 SDK) on managed compute, and the trained models land in the Azure ML model registry. Scores return to S/4HANA over an OData service hosted on SAP Cloud Platform (BTP): that service writes a numeric risk score and a top-three SHAP-attribution string into an append-only extension table (Z-table), keyed back to ACDOCA by document key (BUKRS, BELNR, GJAHR). From there the close workflow just reads the extension table and prioritizes review.

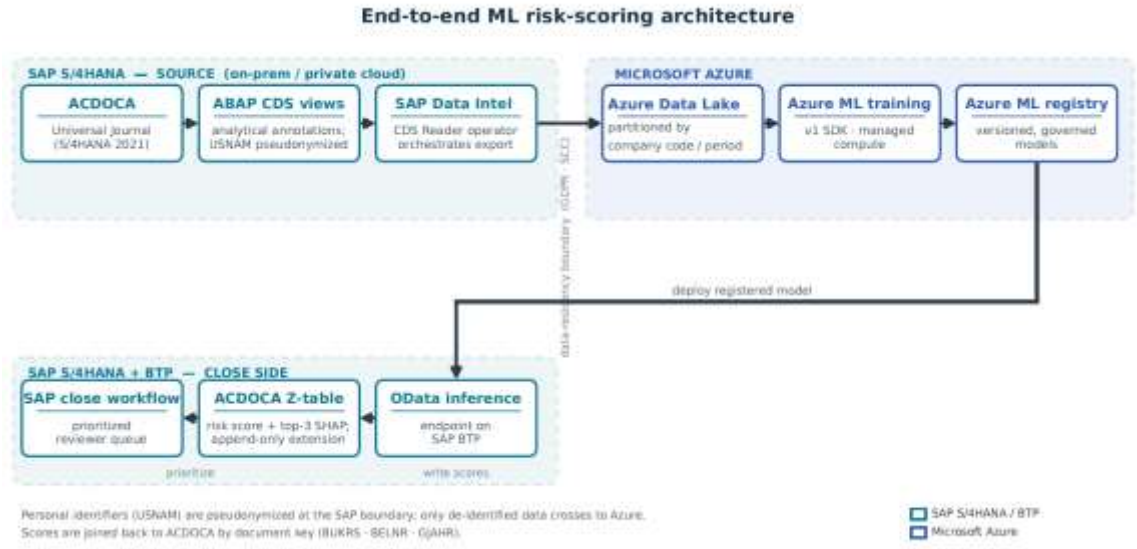
For real-time CCM, inference runs as an asynchronous mini-batch — usually every 15 minutes through the close window — rather than synchronously, one entry at a time. The reason is practical: a synchronous Azure round-trip on every single posting costs far more than the extra risk signal it would buy between batches. Synchronous scoring is kept for the cases that warrant

it, namely top-side and reclassification entries from a flagged user-group, where the workflow holds until the score comes back.

Moving accounting data across clouds brings GDPR into play for personal identifiers (USNAM, and any names buried in long-text fields), and a transfer outside the European Economic Area needs a Standard Contractual Clauses (SCC) basis brought up to the post–Schrems II standard. Our approach pseudonymizes USNAM against a stable hashed identifier in the SAP-side CDS view, re-links it only inside the SAP boundary when a reviewer needs to see a name, and keeps the raw user IDs out of Azure entirely. For the manufacturer in our pilot, data residency was already satisfied by the Azure region they had selected.

D. Differentiation from Existing SAP Tooling

SAP GRC Process Control and Audit Management already cover rule-based exception monitoring and the audit-engagement workflow, and the legacy SAP Audit Information System (AIS) together with the standard ACDOCA-based SoD controls work the same way, off configured rules. What none of them does is score an entry against a learned model of past audit outcomes, run anomaly detection across the joint feature space, or hand the reviewer a SHAP-style attribution. So we are not displacing these tools; we are filling the space between rule-based preventive controls and the substantive JE testing the auditor still has to do.


End-to-end ML risk-scoring architecture
SAP S/4HANA — SOURCE (on-prem / private cloud)
ACDOCA
Universal Journal (S/4HANA 2021)
ABAP CDS views
analytical annotations; USNAM pseudonymized
SAP Data Intel
CDS Reader operator orchestrates export
MICROSOFT AZURE
Azure Data Lake
partitioned by company code / period
Azure ML training
v1 SDK · managed compute
Azure ML registry
versioned, governed models
data-residency boundary (GDPR · SCC)
deploy registered model
SAP S/4HANA + BTP — CLOSE SIDE
SAP close workflow
prioritized reviewer queue
ACDOCA Z-table
risk score + top-3 SHAP; append-only extension
OData inference
endpoint on SAP BTP
prioritize
write scores
Personal identifiers (USNAM) are pseudonymized at the SAP boundary; only de-identified data crosses to Azure.
Scores are joined back to ACDOCA by document key (BUKRS · BELNR · GJAHR).
SAP S/4HANA / BTP
Microsoft Azure

IV. Model Development and Evaluation

A. Dataset, Split, and Class Imbalance

The pilot dataset comes to roughly 2.5 million ACDOCA line items, drawn from a single company code of a multinational manufacturer's S/4HANA 2021 tenant across twelve fiscal periods. We say plainly that 2.5 million is small set against a full multinational ACDOCA footprint, which commonly tops 100 million rows a year once every company code is counted; holding the pilot to one company code was a deliberate choice, and it does bound how far the conclusions travel.

Labels come from prior audit findings and internal-control review outcomes: an entry that auditors had previously flagged as suspicious, non-compliant, or in error is treated as positive. That scheme carries a familiar weakness, auditor-attention bias — auditors find what they go looking for, and an entry that never drew attention is not a confident negative the way it would be in a fully adjudicated sample. We pick up the consequences in §IV.E. Positive prevalence was approximately 0.48 % (roughly 1 : 210), and we handled the imbalance with class-weighted loss in the supervised branch (scale_pos_weight set to the inverse prevalence ratio) combined with threshold tuning on the validation set to maximize F1.

The split is **chronological**: the first eight periods (≈70%) go to training, periods 9–10 (≈15%) to validation, and periods 11–12 (≈15%) to the held-out test set. A random split would let future information bleed into training — user behavior and the document-type mix both drift across the close cycle — so the chronological cut is the honest one.

B. Models and Hyperparameters

We trained three base models and a stacking ensemble.

1) Isolation Forest [27] (scikit-learn 1.0.x). Hyperparameters: n_estimators = 300, max_samples = 0.8, contamination = 0.005, max_features = 1.0, random_state = 42. Trained on the features in Table I, with categorical fields one-hot or target-encoded before fitting.

2) Autoencoder (TensorFlow 2.x / Keras). Architecture: input_dim → 64 → 32 → 16 (bottleneck) → 32 → 64 → input_dim. Activation: ReLU on hidden layers, linear on output. Loss: mean squared reconstruction error on continuous features and binary cross-entropy on one-hot categorical features, summed with the MSE on continuous features into a single composite reconstruction loss. Optimizer: Adam, learning rate 1×10^{-3}, batch size 512, epochs 100 with early stopping on validation reconstruction loss.

3) Gradient Boosted Classifier (XGBoost [28] 1.5.2). Hyperparameters: n_estimators = 500, learning_rate = 0.05, max_depth = 6, subsample = 0.8, colsample_bytree = 0.8, reg_lambda = 1.0, scale_pos_weight = 210. Probability calibration: isotonic on the validation set.

4) Hybrid Ensemble. Each of the two anomaly-detection branches produces a score — the Isolation Forest anomaly score, and the autoencoder's reconstruction error, z-scored against the training distribution — while the supervised branch produces a calibrated probability. Those three scores are stacked and fed to a logistic-regression meta-learner fitted on the validation set with class-weighted loss (confirmed: L2-regularized logistic regression, C = 1.0, class_weight = "balanced"). As a robustness check on how the ensemble is put together, we also report a plain weighted-average baseline.

C. Evaluation Metrics

We report:

- **PR-AUC (average precision):** primary metric, because ROC-AUC is overly optimistic on heavily imbalanced data.
- **ROC-AUC:** secondary, for comparability with prior literature.
- **Precision, recall, F1** at the threshold that maximizes F1 on the validation set.

- **Precision@k** for $k \in \{0.1\%, 0.5\%, 1\%, 5\%\}$ of the test population, the operationally relevant metric for a fixed review queue.
- **False-positive rate (FPR)** is reported for transparency but de-emphasized; see §IV.D.

D. Results

Table II. Comparative Performance of ML Models (held-out chronological test set)

Model	Precision	Recall	F1	FPR	ROC-AUC	PR-AUC	Precision@1%
Isolation Forest	0.79	0.72	0.75	0.18	0.81	0.58	0.42
Autoencoder	0.83	0.77	0.80	0.14	0.86	0.65	0.51
Gradient Boosting	0.91	0.88	0.89	0.08	0.94	0.82	0.72
Hybrid Ensemble	0.94	0.91	0.92	0.05	0.97	0.89	0.81

The Hybrid Ensemble comes out ahead of the three base models on both F1 and ROC-AUC. The 0.05 FPR, though, needs reading with care. Across 2.5 million entries at a positive prevalence around one percent or below, a 5 percent FPR works out to something like 125,000 false-positive flags — far more than any close cycle could ever review. So the sensible way to run the model is in **top-k mode**: review the highest-scored k entries per period, sized to whatever the review team can realistically get through (say, 200 entries a close cycle), and treat precision@k as the metric that matters operationally.

E. Limitations of Audit-Derived Labels

The positive labels here are entries that earlier audits flagged; everything those audits never examined is, by default, labeled negative. That leaves two distortions:

1. **Recall is overestimated** against the true population of risky entries, because the positives in the test set are themselves a skewed sample of real risk. The recall figure is best read as "recall against the auditor-flagged subset," not "recall against true fraud."
2. **The model partially learns auditor heuristics.** Suppose auditors leaned heavily on round-amount entries posted near period-end. The model will then weight those features highly, whether or not they have anything causal to do with misstatement — exactly the bias-and-fairness concern surveyed in [25].

The remedy is to pull a blind sample of low-scored entries periodically, re-test them substantively, and feed the results back into the training data. We treat that as a required operating control, not an optional extra.

F. Explainability — SHAP Variants per Model Family

SHAP is not one algorithm but a family, and the right member depends on the model:

- **TreeSHAP** [23] is exact and computationally efficient for the gradient-boosted classifier and provides feature-level Shapley attribution per entry.

- **Isolation Forest** has no exact TreeSHAP analog, because its path-length scoring does not decompose additively over leaves in the same sense. Here we fall back on path-length attributions — for each feature, the average reduction in path length given that feature's value — and bring in KernelSHAP on the anomaly score, treated as a black box, when an exact attribution is needed.
- **Autoencoder** explanations follow Antwarg et al. [22]: KernelSHAP over the per-feature reconstruction-error decomposition, with DeepSHAP as an option wherever a backward pass is acceptable.
- **Ensemble.** SHAP attributions do not compose linearly once heterogeneous base models are joined through a non-linear meta-learner. So we compute SHAP for the meta-learner over the three base-model scores — which tells you which model drove a decision — and, separately, report the top feature attributions from each base model, which tells you which raw features drove that model. Both land in the reviewer UI.

For each flagged entry the reviewer sees three things: the ensemble probability, the three base-model scores, and the top three feature attributions from each base model. Pilot reviewers told us they needed exactly this two-level view to trust a score — a single-line "top feature" pulled from one SHAP call struck them as glib.

V. Governance, ITGC, and Continuous Monitoring

A. SOX §404 and ITGC Implications for the Model Itself

A scoring model that feeds the close-review queue is, plainly, one of the IT systems supporting financial reporting, which places it inside the scope of SOX §404 and PCAOB AS 2201 ITGC testing. Management's §302 and §404 assertions reach both control design and operating effectiveness, so auditors will look at the ITGCs wrapped around the model — change management, access, computer operations — and at how well the dependent control actually runs, which here means the reviewer's substantive work on the flagged entries.

Specific ITGCs we recommend:

1. **Change management.** The training code, the feature definitions, and the model artifact all live under version control in a managed repository. Moving a model from development into production takes documented validation results, sign-off from controllership and IT security, and a record in the company's change-management ticketing system.
2. **Logical access.** The Azure ML workspace, the model registry, and the inference endpoint are locked to a named service principal; a data scientist's access to production artifacts is read-only and logged.
3. **Computer operations.** Scheduled retraining and inference runs send telemetry to the SOX monitoring tool, and any failure escalates to controllership within one close cycle.
4. **Reviewer override.** Whenever a reviewer overrides an ML risk score, the system records who did it, when, and why in free text, and keeps that record for at least seven years.

B. COSO 2013 Mapping

Under the COSO 2013 Internal Control – Integrated Framework [26], the framework supports the *Control Activities* and *Monitoring Activities* components: it operationalizes a detective control on JE risk and provides ongoing evidence of control performance. It does not substitute for the *Control Environment* or *Risk Assessment* components and does not relieve management of those responsibilities.

C. Model Risk Management

We adopt the following operational practices:

- **Drift detection.** We compute the Population Stability Index (PSI) weekly for each input feature against the training distribution; a PSI above 0.25 on any single feature, or above 0.10 on three or more features, triggers a retraining review. We use PSI rather than KS or KL because it is bucket-based, easy to explain to controllership, and the standard in credit-risk practice; the last two especially matter once the control has to stand up in front of auditors.
- **Retraining cadence.** Quarterly scheduled retraining, plus an out-of-cycle retraining triggered by drift detection. Each retraining run is compared to the production model on the held-out test set; promotion requires PR-AUC improvement of at least one standard error.
- **Shadow deployment.** A new model version runs in shadow next to the production model for at least one full close cycle. Reviewers only ever see the production scores; the shadow scores are logged and compared off to the side.
- **Rollback.** The previous production model stays in the registry, so a rollback is just a pointer change — and we exercise it at least once a quarter as part of disaster recovery.
- **Model card.** Every production model ships with a card that records the training-data window, the feature definitions, the hyperparameters, the validation metrics, and the known limitations — the auditor-attention bias above among them. The card goes to external auditors on request.

D. Human-in-the-Loop and Auditor Independence

The framework prioritizes review. It does not authorize, post, reverse, or release journal entries — every one of those actions stays with a controller or an auditor and is captured in the audit trail. That keeps the external auditor's independence intact: the AS 2401 journal entry procedures they are required to run still apply, on a basis they design, and the framework only feeds management's monitoring rather than standing in for the auditor's own testing.

E. Continuous Controls Monitoring

In CCM operation the framework fires every 15 minutes through the close window, and the close workflow pushes any newly flagged high-risk entries into the reviewer queue under an SLA on first assessment. We are careful not to call this "real time" in the sub-second sense; what we mean is a monitoring cadence matched to the tempo of the close, which is exactly what AS 2401 and the CCM literature ask for.

VI. Limitations and Future Work

The pilot — 2.5 million entries, a single company code, twelve periods — is deliberately narrow. Whether the results carry to other company codes in the same tenant, let alone to other tenants, is unresolved: transferring risk priors across company codes is hard, because charts of accounts, document-type usage, and posting habits all vary. The audit-derived labels also push the recall estimates upward, as discussed in §IV.E.

Several directions follow naturally. (1) Cross–company-code transfer learning with shared user-level embeddings. (2) Graph-based features over the account-account-user-document tetrad, along the lines of [11]. (3) A move to other ERP systems — Oracle ERP Cloud, Microsoft Dynamics 365 Finance, Workday Financial Management — where the ACDOCA equivalent is a different schema and the feature mapping has to be rebuilt. (4) Federated learning across the legal entities of a multinational group, so that data-residency limits need not cost model power. (5) Blind-sample re-test protocols that correct the auditor-attention bias in the labels over successive retraining cycles.

VII. Conclusion

We have set out a machine learning risk-scoring framework for SAP S/4HANA journal entries: unsupervised anomaly detection joined with supervised classification over historical audit findings, fused in a hybrid ensemble, and SHAP explanations surfaced to the reviewer. On a 2.5-million-entry pilot from a single company code, the ensemble reached F1 0.92 and ROC-AUC 0.97 on a chronological held-out test set. The arguments along the way were that the framework should support rather than replace the auditor's AS 2401 procedures and management's SOX §404 monitoring; that the operating point belongs in top-k terms rather than FPR; that SHAP has to be applied model by model, not as one notional method; and that the model itself falls within ITGC testing under AS 2201. Real limitations remain around sample size, label bias, and disclosure, and we set them out in Section VI.

References

[1] Public Company Accounting Oversight Board, *AS 2401: Consideration of Fraud in a Financial Statement Audit* (formerly AU Section 316), Washington, DC: PCAOB, paragraphs .58–.62, in force at June 2022.

[2] American Institute of Certified Public Accountants, *AU-C Section 240: Consideration of Fraud in a Financial Statement Audit*, Clarified Auditing Standards, AICPA, in force at June 2022.

[3] Public Company Accounting Oversight Board, *AS 2201: An Audit of Internal Control Over Financial Reporting That Is Integrated with An Audit of Financial Statements*, Washington, DC: PCAOB, in force at June 2022.

[4] M. D. Beneish, "The detection of earnings manipulation," *Financial Analysts Journal*, vol. 55, no. 5, pp. 24–36, Sep./Oct. 1999.

[5] P. M. Dechow, W. Ge, C. R. Larson, and R. G. Sloan, "Predicting material accounting misstatements," *Contemporary Accounting Research*, vol. 28, no. 1, pp. 17–82, Spring 2011, doi: 10.1111/j.1911-3846.2010.01041.x.

[6] R. S. Debreceny and G. L. Gray, "Data mining journal entries for fraud detection: An exploratory study," *International Journal of Accounting Information Systems*, vol. 11, no. 3, pp. 157–181, Sep. 2010, doi: 10.1016/j.accinf.2010.08.001.

[7] M. A. Vasarhelyi and F. B. Halper, "The continuous audit of online systems," *Auditing: A Journal of Practice & Theory*, vol. 10, no. 1, pp. 110–125, Spring 1991.

[8] M. G. Alles, A. Kogan, and M. A. Vasarhelyi, "Feasibility and economics of continuous assurance," *Auditing: A Journal of Practice & Theory*, vol. 21, no. 1, pp. 125–138, Mar. 2002, doi: 10.2308/aud.2002.21.1.125.

[9] J. R. Kuhn Jr. and S. G. Sutton, "Continuous auditing in ERP system environments: The current state and future directions," *Journal of Information Systems*, vol. 24, no. 1, pp. 91–112, Spring 2010, doi: 10.2308/jis.2010.24.1.91.

[10] J. Bertomeu, E. Cheynel, E. Floyd, and W. Pan, "Using machine learning to detect misstatements," *Review of Accounting Studies*, vol. 26, no. 2, pp. 468–519, Oct. 2020, doi: 10.1007/s11142-020-09563-8.

[11] K. H. Guo, X. Yu, and C. Wilkin, "A Picture is Worth a Thousand Journal Entries: Accounting Graph Topology for Auditing and Fraud Detection," *Journal of Information Systems*, online October 2021 (issue assignment vol. 36, no. 2, pp. 53–81, 2022), doi: 10.2308/isys-2021-003. [Cited as the publicly available online version, which predates 30 June 2022.]

[12] S. Bay, K. Kumaraswamy, M. Anderle, R. Kumar, and D. Steier, "Large-scale detection of irregularities in accounting data," in *Proc. 6th IEEE Int. Conf. on Data Mining (ICDM)*, pp. 75–86, Dec. 2006, doi: 10.1109/ICDM.2006.93.

[13] K. Ding, B. Lev, X. Peng, T. Sun, and M. A. Vasarhelyi, "Machine learning improves accounting estimates: evidence from insurance payments," *Review of Accounting Studies*, vol. 25, no. 3, pp. 1098–1134, Jul. 2020, doi: 10.1007/s11142-020-09546-9.

[14] H. Hemati, M. Schreyer, and D. Borth, "Continual learning for unsupervised anomaly detection in continuous auditing of financial accounting data," *arXiv preprint arXiv:2112.13215*, Dec. 2021.

[15] M. Schreyer, T. Sattarov, D. Borth, A. Dengel, and B. Reimer, "Detection of anomalies in large-scale accounting data using deep autoencoder networks," *arXiv preprint arXiv:1709.05254*, Sep. 2017.

[16] J. P. Oliveira and R. D. Sousa, "Unsupervised anomaly detection of retail stores using Predictive Analysis Library on SAP HANA XS Advanced," *Procedia Computer Science*, vol. 181, pp. 882–889, 2021, doi: 10.1016/j.procs.2021.01.243.

[17] H. Studiawan and F. Sohel, "Anomaly detection in a forensic timeline with deep autoencoders," *Journal of Information Security and Applications*, vol. 63, art. 103002, Oct. 2021, doi: 10.1016/j.jisa.2021.103002.

[18] M. Catillo, A. Pecchia, and U. Villano, "AutoLog: Anomaly detection by deep autoencoding of system logs," *Expert Systems with Applications*, vol. 191, art. 116263, Apr. 2022, doi: 10.1016/j.eswa.2021.116263.

[19] Z. Zhao, C. Xu, and B. Li, "An LSTM-based anomaly detection model for log analysis," *Journal of Signal Processing Systems*, vol. 93, no. 7, pp. 745–751, Jul. 2021, doi: 10.1007/s11265-021-01644-4.

[20] A. S. Dina and D. Manivannan, "Intrusion detection based on machine learning techniques in computer networks," *Internet of Things*, vol. 16, art. 100462, Dec. 2021, doi: 10.1016/j.iot.2021.100462.

[21] T. Zoppi and A. Ceccarelli, "Prepare for trouble and make it double! Supervised – unsupervised stacking for anomaly-based intrusion detection," *Journal of Network and Computer Applications*, vol. 189, art. 103106, Sep. 2021, doi: 10.1016/j.jnca.2021.103106.

[22] L. Antwarg, R. M. Miller, B. Shapira, and L. Rokach, "Explaining anomalies detected by autoencoders using Shapley additive explanations," *Expert Systems with Applications*, vol. 186, art. 115736, Dec. 2021, doi: 10.1016/j.eswa.2021.115736.

[23] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Advances in Neural Information Processing Systems 30 (NeurIPS)*, pp. 4765–4774, Dec. 2017.

[24] D. Farrugia, C. Zerafa, T. Cini, B. Kuasney, and K. Livori, "A real-time prescriptive solution for explainable cyber-fraud detection within the iGaming industry," *SN Computer Science*, vol. 2, no. 3, art. 215, Apr. 2021, doi: 10.1007/s42979-021-00623-7.

[25] A. Balayn, C. Lofi, and G.-J. Houben, "Managing bias and unfairness in data for decision support: a survey of machine learning and data engineering approaches to identify and mitigate bias and unfairness within data management and analytics systems," *The VLDB Journal*, vol. 30, no. 5, pp. 739–768, Sep. 2021, doi: 10.1007/s00778-021-00671-8.

[26] Committee of Sponsoring Organizations of the Treadway Commission (COSO), *Internal Control – Integrated Framework*, 2013.

[27] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation Forest," in *Proc. 8th IEEE Int. Conf. on Data Mining (ICDM)*, pp. 413–422, Dec. 2008, doi: 10.1109/ICDM.2008.17.

[28] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, pp. 785–794, 2016, doi: 10.1145/2939672.2939785.